Refine Search

Search Results -

Terms	Documents
L53 and network	15

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:

Refine Search

Recall Text

Clear

Interrupt

Search History

DATE: Tuesday, April 27, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L54	L53 and network	15	L54
L53	mirror adj engine	31	L53
L52	714/15	1352	L52
L51	707/204	1753	L51
L50	707/206	876	L50
L49	707/100	4479	L49
L48	L47 and backup near system	48	L48
L47	L46 and network near servers	921	L47
L46	l41 and data near stor$	7659	L46
L45	L44 and mirror same engine	11	L45
L44	L43 and data	2046	L44
L43	L42 and write	2056	L43
L42	L41 and server	5117	L42
L41	mass near stor$ near device	12697	L41

L40	L38 and stor$ near node	27	L40
L39	L38 and polic$ near protocol	1	L39
L38	L37 and (I/O driver or on/off driver)	181	L38
L37	L36 and workstations	253	L37
L36	L34 and servers	808	L36
L35	L34 and shared near stor$ near node	5	L35
L34	L33 and (network or www or internet)	1190	L34
L33	mirror$ near data	2178	L33
L32	709/232	1829	L32
L31	709/214	546	L31
L30	370.clas.	73660	L30
L29	370/417	402	L29
L28	714/6	2251	L28
L27	714/4	2059	L27
L26	714.clas.	44656	L26
L25	711/147	1630	L25
L24	711/111	1200	L24
L23	711/114	2103	L23
L22	707/104.1	3936	L22
L21	707/10	8221	L21
L20	707/9	2086	L20
L19	707/8	1920	L19
L18	707/1	6308	L18
L17	707/201	2171	L17
L16	707/200	3206	L16
L15	711.clas.	21063	L15
L14	707.clas.	20120	L14
L13	709.clas.	28166	L13
L12	709/200	2595	L12
L11	709/229	3512	L11
L10	709/227	3785	L10
L9	709/224	5297	L9
L8	709/212	627	L8
L7	709/217	5121	L7
L6	709/216	801	L6
L5	709/201	3393	L5
L4	709/204	1927	L4
L3	709/203	7203	L3
L2	709/213	1579	L2
L1	709/219	4886	L1

END OF SEARCH HISTORY

First Hit Fwd Refs

Generate Collection Print

L48: Entry 38 of 48 File: USPT Nov 3, 1998

US-PAT-NO: 5832522
DOCUMENT-IDENTIFIER: US 5832522 A

TITLE: Data storage management for network interconnected processors

DATE-ISSUED: November 3, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Blickenstaff; Ronald L.	Niwot	CO		
Brant; Catherine Irlam	Boulder	CO		
Dodd; Paul David	Niwot	CO		
Kirchner; Anton H.	Nederland	CO		
Montez; Jennifer Kay	Thornton	CO		
Trede; Brian Eldred	Woodinville	WA		
Winter; Richard Allen	Longmont	CO		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Kodak Limited	Harrow			GB2	03

APPL-NO: 08/ 650114 [PALM]
DATE FILED: May 22, 1996

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATION This application is a divisional of a patent application entitled "Data Storage Management For Network Interconnected Processors," Ser. No. 08/201,658 filed Feb. 25, 1994, now U.S. Pat. No. 5,537,585.

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 707/204; 707/10, 707/205
US-CL-CURRENT: 707/204; 707/10, 707/205

FIELD-OF-SEARCH: 395/620, 395/621, 395/440, 395/441, 395/444, 395/800, 707/204, 707/205, 707/10

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL

☐	5155835	October 1992	Belsan	395/441
☐	5218695	June 1993	Noveck et al.	395/600
☐	5276860	January 1994	Fortier et al.	395/575
☐	5276867	January 1994	Kenley et al.	395/600
☐	5313631	May 1994	Kao	395/600
☐	5367698	November 1994	Webber et al.	395/800
☐	5581724	December 1996	Belsan et al.	395/441

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 92/09035	May 1992	WO	

OTHER PUBLICATIONS

Proceedings of the Symposium On Mass Storage Systems, Symp. 11, Oct. 17, 1991 Institute of Electrical & Electronics Engineers, pp. 3-10, by Foster et al. Renaissance: Managing the Network Computer and its Storage Requirements.
Data Communications, vol. 22, No. 11, pp. 49-50, , by S. Salamone, "Migrating Data To Cheaper Storage".
Proceedings of the IEEE, vol. 63, No. 8, pp. 1166-1170, by C. Johnson, "The IBM 3850: A Mass Storage System with Disk Characteristics".
"Experience With File Migration" by R. D. Christman, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, reproduced by National Technical Information Service, U.S. Dept. of Commerce, Springfield, VA 22161, Oct. 1981.
"File Migration In The NCAR Mass Storage System" by E. Thanhardt & G. Harano, National Center for Atmospheric Research, Boulder, Colorado, Ninth IEEE Symposium on Mass Storage Systems, 1988.
"A Distributed Algorithm for Performance Improvement Through File Replication, File Migration, and Process Migration" by A. Had, IEEE Transactions on Software Engineering vol. 15, Issue 11, pp. 1459-1470, Nov. 1989.
"Development of Omniserver" by. D. A. Arneson, Control Data Corporation, Minneapolis, MN, 1990 IEEE, pp. 88-93.
"Architecture And Implementation of an On-line Data Archive and Distribution System", by. B. Bhasker, M. E. Van Steenberg, B.E. Jacobs, Hughes STX Corp., Lanham, MD, Proceedings Twelfth IEEE Symposium on Mass Storage Systems. Cat. No. 93CH3246-6, pp. 177-182, Apr. 1993.
"Potential Benefits of File Migration in a Heterogeneous Distributed File Systems" by R. T. Hurley, S.A. Yeap, J. W. Wong, J.P. Black, Proceedings ICCI '93, Fifth International Conference on Computing and Information, Cat. No. 93Th0563-7, pp. 123-127, May 1993.

ART-UNIT: 271

PRIMARY-EXAMINER: Lintz; Paul R.

ATTY-AGENT-FIRM: Duft, Graziano & Forest, P.C.

ABSTRACT:

The data storage system is connected to a local area network and includes a storage server that on a demand basis and/or on a periodically scheduled basis audits the

activity on each volume of each data storage device that is connected to the network. Low priority data files are migrated via the network and the storage server to backend data storage media, and the directory resident in the data storage device is updated with a placeholder entry to indicate that this data file has been migrated to backend storage. When the processor requests this data file, the placeholder entry enables the storage server to recall the requested data file to the data storage device from which it originated.

25 Claims, 13 Drawing figures

First Hit Fwd Refs

☐ Generate Collection Print

L48: Entry 40 of 48 File: USPT Sep 1, 1998

US-PAT-NO: 5802366
DOCUMENT-IDENTIFIER: US 5802366 A

TITLE: Parallel I/O network file server architecture

DATE-ISSUED: September 1, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Row; Edward John	Mountain View	CA		
Boucher; Laurence B.	Saratoga	CA		
Pitts; William M.	Los Altos	CA		
Blightman; Stephen E.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Auspex Systems, Inc.	Santa Clara	CA			02

APPL-NO: 08/ 320451 [PALM]
DATE FILED: October 11, 1994

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS This application is a continuation of U.S. patent application Ser. No. 07/959,746, filed Oct. 13, 1992, now U.S. Pat. No. 5,355,453, which is a continuation of U.S. patent application Ser. No. 07/404,959, filed Sep. 8, 1989, now U.S. Pat. No. 5,163,131. 1. MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE, invented by David Hitz, Allan Schwartz, James Lau and Guy Harris; 2. ENHANCED VMEBUS PROTOCOL UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER, invented by Daryl Starr; and 3. BUS LOCKING FIFO MULTI-PROCESSOR COMMUNICATIONS SYSTEM UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER invented by Daryl D. Starr, William Pitts and Stephen Blightman. The above applications are all assigned to the assignee of the present invention and are all expressly incorporated herein by reference.

INT-CL: [06] G06 F 13/00

US-CL-ISSUED: 395/683
US-CL-CURRENT: 709/250

FIELD-OF-SEARCH: 395/200.02, 395/200.06, 395/200.12, 395/700, 395/650, 395/680, 395/683

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL | Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4075691	February 1978	Davis et al.	364/200
☐	4156907	May 1979	Rawlings et al.	364/200
☐	4333144	June 1982	Whiteside et al.	364/200
☐	4377843	March 1983	Garringer et al.	364/200
☐	4399503	August 1983	Hawley	364/200
☐	4456957	June 1984	Schieltz	364/200
☐	4459664	July 1984	Pottier et al.	364/200
☐	4488231	December 1984	Yu et al.	364/200
☐	4527232	July 1985	Bechtolsheim	364/200
☐	4550368	October 1985	Bechtolsheim	364/200
☐	4685125	August 1987	Zave	364/200
☐	4710868	December 1987	Cocke et al.	364/200
☐	4719569	January 1988	Ludemann et al.	364/200
☐	4766534	August 1988	De Benedictis	364/200
☐	4780821	October 1988	Crossley	364/200
☐	4783730	November 1988	Fischer	364/200
☐	4803621	February 1989	Kelly	364/200
☐	4819159	April 1989	Shipley et al.	364/200
☐	4825354	April 1989	Lighthart et al.	364/200
☐	4887204	December 1989	Johnson et al.	364/200
☐	4897781	January 1990	Chang et al.	364/200
☐	4914583	April 1990	Weisshaar et al.	364/200

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 321 723 A2	June 1989	EP	
WO 89/03086	April 1989	WO	

OTHER PUBLICATIONS

Frank, Werner L., "Share a Common Storage Device Via File Servers", PC Week (Mar. 20, 1984), p. 19.
Jenkins, Avery, "Choosing the Elements of Your Network", PC Week (Mar. 13, 1984), pp. 28-29, 35.
Kramer, Matt, "MS-NET Debut Does Not Deter LAN Vendors", PC Week (Nov. 20, 1984), p. 11.
Tanenbaum, Andrew S., "Computer Networks" (1988), 2nd Edition, Prentice Hall, pp. 35, 36, Chap. 9.

IEEE Computer, "I/O subsystem", Sep. 1988, pp. 23-25 and 106.
Carlson, et al., "HP AdvanceNet: A Growth-Oriented Computer Networking Architectural Strategy", Hewlett-Packard Journal (Oct. 1986), p. 2, pp. 6-10.
Tribby, David M., "Network Services for HP Real-Time Computers", Hewlett-Packard Journal (Oct. 1986), pp. 22-27.
Motorola, Inc., Microsystems Products Technical Data Sheet (1986), microMAP1-7, "MicroMAP Manufacturing Automation Protocol Software".
Osadzinski, Alex, "The Network File System (NFS)", 8202 Computer Standards & Interfaces, 8 (1988/89) No. 1, pp. 45-48, Amsterdam, The Netherlands.
"TCP/IP Software", PC Week (Connectivity section) (Mar. 8, 1988), p. C/7.
Birkhead, Evan, "Solutions Connect VMS To Host-Based TCP/IP And NFS", DEC Professional (Feb. 1989), vol. 6, Issue 14, p. 28.
Birkhead, Evan, "Topic's Connectivity Scheme", DEC Professional (May 1989), vol. 8, Issue 5, p. 28.
Breidenbach, Susan, "Novell joins supporters of Sun's NFS", Network World (Feb. 20, 1989), vol. 6, Issue 7, p. 5.
Breidenbach, Susan, "Start-up unveils accelerator for boosting speed of NFS", Network World (Jun. 12, 1989), vol. 6, Issue 23, pp. 19-20.
Eckerson, Wayne, "Sun, EDS Shaping Network File System for IBM's MVS", Network World (Sep. 19,1988), vol. 5, Issue: 38, pp. 2,59.
Keefe, Patricia, "Sun fills out IBM connectivity line", Computerworld (Sep. 19, 1988), p. 16.
Musich, Paula, "MVS Version of Sun NFS Software Announced", PC Week (Connectivity section) (Sep. 19, 1988), vol. 22, Issue 38 p. C/3.
Musich, Paula, "CMC Tailors OpenWare for DEC and Unix Environments", PC Week (Connectivity section) (Aug. 28, 1989), vol. 6, Issue 34, p. 53.
Row, John, "LAN Software Links Diverse Machines, OS's", Mini-Micro System (Sep. 1985), pp. 141-142, 145, 147.
Row, et al., "Operating System Extensions Link Disparate Systems", Computer Designs (Jul. 1984).
Sandberg, Russel, "The Sun Network Filesystem: Design, Implementation and Experience", Proceedings of the 1986 European Unix User's Conference (Apr. 1986).
Schnatmeier, Vanessa, "Epoch Delivers Mass Magnetic/Optical Storage for Workstations", UnixWorld (Dec. 1988), p. 134.
Siegel, Alex, et al., "Deceit: A Flexible Distributed File System", National Aeronautics and Space Administration, Report No.: NAS 1.26:186412; TR-89-1042; NASA-CR-186412 (Dec. 7, 1989).
Smalley, Eric, "CMC Adds Support for NFS to VAX", Digital Review (Aug. 28, 1989), pp. 1,6.
Sorensen, Kristiina, "Sun's NFS Circle Widens To Include MVS Mainframes", Digital Review (Sep. 26, 1988), pp. 1, 102.
Sullivan, Kristina B., "DEC Extends High End of VAX-Based File-Server Line", PC Week (Dec. 12, 1988), vol. 5, Issue 50, pp. C/4, C/8.
Sun Microsystems, Inc., NFS: Network File System Protocol Specification, Request for Comments: 1094 (Mar. 1989).
Tan, S.M., et al., "SOS -Stan's Own Server, A NFS file server for the IBM PC", Department of Energy Report No. LBL-25770 (Aug. 1988).
Vizard, Michael, "VAX TCP/IP Networking Option Will Support NFS Software That Runs on Sun", Digital Review (Nov. 23, 1987).
Vizard, Michael, "1988 Sees DEC Take a New Tack: Won't Go It Alone", Digital Review (Dec. 19, 1988), vol. 5, Issue 24, pp. 8, 87.

ART-UNIT: 236

PRIMARY-EXAMINER: Kriess; Kevin A.

ATTY-AGENT-FIRM: Fliesler, Dubb, Meyer & Lovejoy

ABSTRACT:

A file server architecture is disclosed, comprising as separate processors, a network controller unit, a file controller unit and a storage processor unit. These units incorporate their own processors, and operate in parallel with a local Unix host processor. All networks are connected to the network controller unit, which performs all protocol processing up through the NFS layer. The virtual file system is implemented in the file control unit, and the storage processor provides high-speed multiplexed access to an array of mass storage devices. The file controller unit control file information caching through its own local cache buffer, and controls disk data caching through a large system memory which is accessible on a bus by any of the processors.

20 Claims, 12 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L48: Entry 43 of 48 File: USPT Mar 4, 1997

US-PAT-NO: 5608865
DOCUMENT-IDENTIFIER: US 5608865 A

TITLE: Stand-in Computer file server providing fast recovery from computer file server failures

DATE-ISSUED: March 4, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Midgely; Christopher W.	Framingham	MA		
Holland; Charles	Northboro	MA		
Holberger; Kenneth D.	Grafton	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Network Integrity, Inc.	Marlborough	MA			02

APPL-NO: 08/ 405178 [PALM]
DATE FILED: March 14, 1995

INT-CL: [06] G06 F 11/00

US-CL-ISSUED: 395/180; 395/182.04, 395/616
US-CL-CURRENT: 714/1; 707/200, 714/6

FIELD-OF-SEARCH: 395/180, 395/181, 395/182.04, 395/182.05, 395/182.8, 395/600, 395/650

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5151989	September 1992	Johnson et al.	395/600
☐	5210866	May 1993	Milligan et al.	395/182.17
☐	5369757	November 1994	Spiro et al.	395/182.17
☐	5410691	April 1995	Taylor	395/600
☐	5459863	October 1995	Taylor	395/600

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO94/17473	August 1994	WO	
WO94/17474	August 1994	WO	

ART-UNIT: 243

PRIMARY-EXAMINER: Beausoliel, Jr.; Robert W.

ASSISTANT-EXAMINER: Hua; Ly V.

ATTY-AGENT-FIRM: Fish & Richardson P.C.

ABSTRACT:

An Integrity Server computer for economically protecting the data of a computer network's servers, and providing hot standby access to up-to-date copies of the data of a failed server. As the servers' files are created or modified, they are copied to the Integrity Server. When one of the servers fails, the Integrity Server fills in for the failed server, transparently providing the file service of the failed server to network clients. The invention provides novel methods for managing the data stored on the Integrity Server, so that the standby files are stored on low-cost media such as tape, but are quickly copied to disk when a protected server fails. The invention also provides methods for re-establishing connections between clients and servers, and communicating packets between network nodes, to allow the Integrity Server to stand-in for a failed server without requiring reconfiguration of the network clients.

6 Claims, 13 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L48: Entry 44 of 48 File: USPT Feb 18, 1997

US-PAT-NO: 5604862
DOCUMENT-IDENTIFIER: US 5604862 A

TITLE: Continuously-snapshotted protection of computer files

DATE-ISSUED: February 18, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Midgely; Christopher W.	Framingham	MA		
Holland; Charles J.	Northboro	MA		
Webb; John W.	Sutton	MA		
Gonsalves; Manuel	Brookline	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Network Integrity, Inc.	Marlborough	MA			02

APPL-NO: 08/ 403347 [PALM]
DATE FILED: March 14, 1995

INT-CL: [06] G06 F 11/34

US-CL-ISSUED: 395/182.04; 395/488, 395/859
US-CL-CURRENT: 714/6; 710/39, 711/161

FIELD-OF-SEARCH: 395/181, 395/182.03, 395/182.04, 395/182.05, 395/427, 395/444, 395/445, 395/859, 395/488, 395/489, 371/7

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4310883	January 1982	Clifton et al.	364/200
☐	5086502	February 1992	Malcolm	364/200
☐	5163148	November 1992	Walls	364/200
☐	5210866	May 1993	Milligan et al.	364/200
☐	5239647	August 1993	Anglin et al.	364/200
	5247660	September 1993	Ashcraft et al.	364/900

☐				
☐	5263154	November 1993	Eastridge et al.	364/268.2
☐	5276860	January 1994	Fortier et al.	364/200
☐	5276867	January 1994	Kenley et al.	395/600
☐	5367698	November 1994	Webber et al.	364/200
☐	5386545	January 1995	Gombos, Jr. et al.	395/700
☐	5403639	April 1995	Belsan et al.	395/600

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO94/17473	August 1994	WO	
WO94/17474	August 1994	WO	

OTHER PUBLICATIONS

ProServe CX "NLM Backup for Netware" Installation Guide, 1994.
Networker, Users Guide, 1994.
ProServe CX "NLM Backup for Netware", Administrator's Guide, 1994.
Networker, Administrator's Guide, 1994.
White Paper, "St. Bernard Software Open File Manager", Mar. 7, 1995.

ART-UNIT: 243

PRIMARY-EXAMINER: Beausoliel, Jr.; Robert W.

ASSISTANT-EXAMINER: Decady; Albert

ATTY-AGENT-FIRM: Fish & Richardson P.C.

ABSTRACT:

An Integrity Server computer for economically protecting the data of a computer network's servers, and providing hot standby access to up-to-date copies of the data of a failed server. As the servers' files are created or modified, they are copied to the Integrity Server. The invention provides novel methods for managing the data stored on the Integrity Server, so that up-to-date snapshots of files of the protected file servers are stored on low-cost media such as tape, but without requiring that a system manager load large numbers of tapes.

31 Claims, 5 Drawing figures

L48: Entry 46 of 48 File: USPT Oct 11, 1994

US-PAT-NO: 5355453
DOCUMENT-IDENTIFIER: US 5355453 A
** See image for Certificate of Correction **

TITLE: Parallel I/O network file server architecture

DATE-ISSUED: October 11, 1994

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Row; Edward J.	Mountain View	CA		
Boucher; Laurence B.	Saratoga	CA		
Pitts; William M.	Los Altos	CA		
Blightman; Stephen E.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Auspex Systems, Inc.	Santa Clara	CA			02

DISCLAIMER DATE: 20091110

APPL-NO: 07/ 959746 [PALM]
DATE FILED: October 13, 1992

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS This application is a continuation of U.S. patent application Ser. No. 07/404,959, filed Sep. 8, 1989 now U.S. Pat. No. 5,163,131. The present application is related to the following U.S. Patent Applications, all filed concurrently herewith: 1. MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE, invented by David Hitz, Allan Schwartz, James Lau and Guy Harris; 2. ENHANCED VMEBUS PROTOCOL UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER, invented by Daryl Starr; and 3. BUS LOCKING FIFO MULTI-PROCESSOR COMMUNICATIONS SYSTEM UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER invented by Daryl D. Starr, William Pitts and Stephen Blightman. The above applications are all assigned to the assignee of the present invention and are all expressly incorporated herein by reference.

INT-CL: [05] G06F 15/16, G06F 13/00

US-CL-ISSUED: 395/200; 364/DIG.1, 364/242.4, 364/228.3, 364/234, 364/284.4, 364/243.4, 364/230
US-CL-CURRENT: 709/219; 709/212

FIELD-OF-SEARCH: 395/200, 395/650, 395/700, 395/800

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL | Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4075691	February 1978	Davis et al.	364/200
☐	4156907	April 1979	Rawlings et al.	364/200
☐	4333144	June 1982	Whiteside et al.	364/200
☐	4377843	March 1983	Garringer et al.	364/200
☐	4399503	August 1983	Hawley	364/200
☐	4456457	June 1984	Schieltz	364/200
☐	4459664	July 1984	Pottier et al.	364/200
☐	4488231	December 1984	Yu et al.	364/200
☐	4527232	July 1985	Bechtolsheim	364/200
☐	4550368	October 1985	Bechtolsheim	364/200
☐	4685125	August 1987	Zave	364/200
☐	4710868	December 1987	Cocke et al.	364/200
☐	4719569	January 1988	Ludemann et al.	364/200
☐	4766534	August 1988	DeBenedictis	364/200
☐	4780821	October 1988	Crossley	364/200
☐	4783730	November 1988	Fischer	364/200
☐	4803621	February 1989	Kelly	364/200
☐	4819159	April 1989	Shipley et al.	364/200
☐	4825354	July 1989	Lighthart et al.	364/200
☐	4887204	December 1989	Johnson et al.	364/200
☐	4897781	January 1990	Chang et al.	364/200
☐	4914583	April 1990	Weisshaar et al.	364/200

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0321723A2	June 1989	EP	
WO 89/03086	April 1989	WO	

OTHER PUBLICATIONS

Osadzinski, Alex, "The Network File System (NFS)", 8202 Computer Standards & Interfaces, 8 (1988/89) No. 1, pp. 45-48, Amsterdam, The Netherlands.
Tannebaum, Andrew S., "Computer Networks" (1988), 2nd Edition, Prentice Hall, pp. 35, 36, Chap. 9.
IEEE Computer, "I/O subsystem", Sep. 1988, pp. 23-25 and 106.
Carlson, et al., "HP AdvanceNet: A Growth-Oriented Computer Networking

Architectural Strategy", Hewlett-Packard Journal (Oct. 1986), p. 2, pp. 6-10.
Tribby, David M., "Network Services for HP Real-Time Computers", Hewlett-Packard Journal (Oct. 1986), pp. 22-27.
Motorola, Inc., Microsystems Products Technical Data Sheet (1986), microMAP1-7, "MicroMAP Manufacturing Automation Protocol Software".

ART-UNIT: 236

PRIMARY-EXAMINER: Kriess; Kevin A.

ATTY-AGENT-FIRM: Fliesler, Dubb, Meyer & Lovejoy

ABSTRACT:

A file server architecture is disclosed, comprising as separate processors, a network controller unit, a file controller unit and a storage processor unit. These units incorporate their own processors, and operate in parallel with a local Unix host processor. All networks are connected to the network controller unit, which performs all protocol processing up through the NFS layer. The virtual file system is implemented in the file control unit, and the storage processor provides high-speed multiplexed access to an array of mass storage devices. The file controller unit control file information caching through its own local cache buffer, and controls disk data caching through a large system memory which is accessible on a bus by any of the processors.

38 Claims, 12 Drawing figures

First Hit

☐ Generate Collection Print

L54: Entry 6 of 15 File: PGPB Nov 1, 2001

PGPUB-DOCUMENT-NUMBER: 20010037371
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20010037371 A1

TITLE: Mirroring network data to establish virtual storage area network

PUBLICATION-DATE: November 1, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Ohran, Michael R.	Orem	UT	US	

APPL-NO: 09/ 892161 [PALM]
DATE FILED: June 26, 2001

RELATED-US-APPL-DATA:
Application 09/892161 is a continuation-in-part-of US application 09/271585, filed March 18, 1999, PENDING
Application 09/271585 is a continuation-of US application 08/848139, filed April 28, 1997, US Patent No. 5978565

INT-CL: [07] G06 F 15/167

US-CL-PUBLISHED: 709/214; 709/232
US-CL-CURRENT: 709/214; 709/232

REPRESENTATIVE-FIGURES: 1

ABSTRACT:

Mirroring data to provide a virtual storage area network using policing protocols and mirror engines without a physical shared storage node. The mirror engines are found at each server computer in the network in order to mirror the data between mass storage devices of the servers as the servers receive and execute write operations, which results in each mass storage device containing the same stored data. The policing protocols prevent data corruption by not allowing more than one server at a time write to a file of data. If one server experiences failure and is incapable of providing access to network data, the other server or servers can service all read requests, since all network data is accessible by all servers. Unlike conventional storage area networks, there is no physical shared storage node and, accordingly, the costs of obtaining and operating the virtual storage area network are relatively small.

RELATED APPLICATIONS

[0001] This application is a continuation-in-part of U.S. patent application Ser. No. 09/271,585, entitled "Operation of Standby Server to Preserve Data Stored By a

Network Server," filed Mar. 18, 1999, which is a continuation of U.S. patent application Ser. No. 08/848,139, filed Apr. 28, 1997, entitled "Method for Rapid Recovery from a Network File Server Failure Including Method for Operating Co-Standby Servers," now issued as U.S. Pat. No. 5,978,565. The foregoing patent and patent application are incorporated herein by reference.

First Hit

Generate Collection Print

L54: Entry 4 of 15 File: PGPB Nov 28, 2002

DOCUMENT-IDENTIFIER: US 20020178264 A1
TITLE: Secure creation and distribution of instructions to uniquely support network applications

Abstract Paragraph:
A system, method and computer program product for the secure creation and distribution of instructions to support network applications is provided. The system allows nodes within a network to transform from the functionality of a "client" into a "server" with no residual legacy information being discernable by the new function and vice-versa. The method and product function as a browser, built directly onto the network's transport-network layer protocol (e.g., TCP/IP) that does not require pre-defined formats for the presentation of any page, segment, unit, or quantity of information, and without regard as to the information type or category. Therefore, the present invention serves as a network operating platform (or engine) for service providers to offer network-based (e.g., Internet-based) application programs to their clients.

Summary of Invention Paragraph:
[0002] The present invention relates generally to computer systems, and more particularly to computer operating platforms that support and facilitate distributed network-based application programs.

Summary of Invention Paragraph:
[0004] In today's technological climate, the availability of low-cost yet powerful computers, networking equipment, telecommunications, and related technology has dramatically changed the way people communicate. That is, as computers have become more powerful and ubiquitous, it has become important to connect them together in networks to facilitate communication among them (and, more importantly, their users). For example, the explosion of people connected to the global (sometimes referred to as the "public") Internet has dramatically increased the usage of electronic mail (e-mail) for communications, and the use of the browsers to navigate between and view (i.e., browse) documents through the World-Wide Web.

Summary of Invention Paragraph:
[0005] Generically speaking, a network is comprised of "nodes" and "links." Each node is simply a computation device (e.g., whether personal computer, gateway, router and the like) connected to the network via a communications link. The links, which allow data to be communicated among the several nodes of a computer network, are such that there exists a path from each node over the links and possibly through the other nodes to each of the other nodes in the network. That is, each node may be connected to the network with one or more links. Networks are typically classified according to their geographic extent (e.g., wide-area network or local-area network) and the protocol (i.e., set of formal rules describing how to transmit data across the network) used to communicate among the nodes.

Summary of Invention Paragraph:
[0006] The connectivity achieved by the Internet--connecting a great number of computers within different types of networks--is based upon a common protocol suite utilized by those computers connecting to it. Part of the common protocol suite is the Internet Protocol (IP), defined in Internet Standard (STD) 5, Request for

Comments (RFC) 791 (Internet Architecture Board). IP is a network layer (in terms of the International Organization for Standardization (ISO) Open Systems Interconnect (OSI) seven-layer model), packet (i.e., a unit of transmitted data) switching protocol. IP uses address to distinguish among the computers in a network or more specifically, to distinguish among the millions of computers connected to the global Internet. An IP address is specified by a 32-bit host address usually represented in dotted decimal notation (e.g. 128.121.3.5). The IP address format is currently defined in STD 5, RFC 791.

Summary of Invention Paragraph:
[0008] Most typically, application programs running on computer networks are implemented and described in terms of the "client-server" paradigm. That is, nodes within a computer network are classified as either a "client" or a "server." The client-server paradigm and the use of the TCP/IP protocol suite is described in detail in Douglas E. Comer & David L. Stevens, "Internetworking with TCP/IP: Vol. III Client--Server Programming and Applications," ISBN 0-13-474230-3, Prentice Hall (USA 1994), which is incorporated herein by reference in its entirety.

Summary of Invention Paragraph:
[0010] A "server" is a node running a process that provides some process or information to other computers (or nodes) connected to it via a network. The most common example is a computer file server that has a local storage disk, and services requests from remote client computers to read and write files on that disk.

Summary of Invention Paragraph:
[0012] Commercially speaking, it is most typical to have an application service provider (ASP) that provides and allows access, perhaps on a subscriber basis or per-use basis, to an application program via the Internet. That is, the ASP would provide the hardware (e.g., server machines) and software (e.g., application software and data stores) infrastructure to allow its customers (e.g., users using commercially available Web browser software) to execute their offered application software. Such application programs typically include Web site offerings such as electronic auction, Internet search engines, entertainment or recreational games, business-to-business tools, chat services, e-mail services, personal financial services, information or news services, data retrieval services, product offerings or the like. Further, the type of application program offered by the ASP dictates the security concerns of data passing between nodes within the network. This is the conventional model.

Summary of Invention Paragraph:
[0013] In a computer network setting, the role of either the client or the server does not change from its defined role of the former at the outset of the computational process, to a new role as the latter, while the computational process is still underway. Similarly, the role of a computational process does not change from the function of a server to that of a client, while the computational process is underway.

Summary of Invention Paragraph:
[0014] At the present time, all Internet browser applications suffer from diminished efficiencies, because the present embodiments of software necessarily treat each computational node on a network as either a "client" only, or a "server." This rigid "client-server" model does not lend itself well to various tasks, especially in computations using time-sensitive dynamic variables. These dynamic variables may be required to assess situations that are of short duration, or transitive in nature.

Summary of Invention Paragraph:
[0016] As a direct result of these present inefficiencies, both bandwidth resources and timely information are either compromised or unavailable. Further, within the

client-server paradigm, and more specifically within Web browser applications utilizing TCP/IP within the Internet, ASPs are limited in their ability to provide maximum computational power. That is, there is a need for a process engine that removes the structure of a client and a server in the network computational model . With such an engine (or Internet operating platform) in place, application program instructions may be passed from one point to another in an optimized format, and can be executed in a location that may be more efficient for a given process.

Summary of Invention Paragraph:
[0017] Therefore, what is needed is a system, method and computer program product for the secure creation and distribution of instructions to support (Internet) applications that can be executed anywhere in a network. The system, method and product should also comprise a presentation means (e.g., browser) that does not require pre-defined formats for the presentation of any page, segment, unit, or quantity of information, without regard as to its type or category. The system, method, and product should also allow for the presentation of data that can be unique to each of a multiplicity of users, based upon each user's needs at that instant.

Summary of Invention Paragraph:
[0018] The present invention meets the above-mentioned needs by providing a system, method and computer program product for the secure creation and distribution of instructions to support network (e.g., Internet) applications. The creation and presentation of these instructions or data can be unique to any node to which it is presented. The system, method and computer program product of the present invention functions as a process engine (e.g., a network operating platform) that removes the structure of a client and a server in the typical network computational model.

Summary of Invention Paragraph:
[0019] The system of the present invention includes a dispatcher machine connected over a network (e.g., the Internet) to several user machines, wherein the dispatcher has means for receiving a connection request from the one of the user machine. The system also includes one or more databases connected to the dispatcher and a cluster of servant nodes. These databases include computational resource data, security information, subscriber information, system status information and application program specific data. The dispatcher, databases and servant machines make up an "engine" or platform that facilitates an ASP's delivery of network-based application programs to its subscribers (i.e., users or customers).

Summary of Invention Paragraph:
[0022] The method and product then commences an application program on the selected servant node. The servant node, in providing the application program to the user, will receive a query from the user, wherein the query is related to the application program. The servant node will then send a second code package to the user machine, wherein the second code package includes instructions and data which may be completely or partially responsive to the query. If partially responsive to the query, the servant machine instructions direct the user machine to execute the instructions (i.e., executable code) and data received from servant node and to then forward a computed answer back to the servant node. The servant machine then parses the answer received from the user machine and combines it with additional data to form a final response to the query received from the user machine. If completely responsive, the servant machine computes the final response. Thus, a final response is formed and then sent to the user machine, whereby the first and second code packages facilitate an ASP's delivery of a network-based application program to a user.

Summary of Invention Paragraph:
[0023] One advantage of the present invention is that it allows computational operations to be performed more efficiently, through reduced network transit latency, by requiring less information to transit the network. The present

invention also further decreases network latencies by calculating the expected datagram sizes, and executing such optimizing techniques as "expedited forwarding", dynamically adjusting maximum transmission unit (MTU) window sizes, or other means which are known to those skilled in the relevant art(s).

Summary of Invention Paragraph:
[0024] Another advantage of the present invention is that it enables each computational node (i.e., addressable computer device) connected to a network to receive computer-determined instructions that are unique to each computational node.

Summary of Invention Paragraph:
[0025] Another advantage of the present invention is that it allows nodes within a network to transform from the functionality of a "client" into a "server" with no residual legacy information being discernable by the new function and vice-versa.

Summary of Invention Paragraph:
[0026] Yet another advantage of the present invention is that it enables scalability that has dramatically fewer restrictions over conventional network-based application provision techniques where client and server functions (roles) are fixed.

Brief Description of Drawings Paragraph:
[0031] FIG. 2 is a block diagram illustrating the physical architecture of a network operating platform system, according to an embodiment of the present invention, showing connectivity among the various components;

Brief Description of Drawings Paragraph:
[0032] FIG. 3 is a flow chart depicting an embodiment of the operation and control flow of a user utilizing an application program implemented using the process engine (i.e., network operating platform) of the present invention;

Brief Description of Drawings Paragraph:
[0033] FIG. 4 is a block diagram of the software architecture of part of the process engine (i.e., network operating platform) that resides on the servant machines, according to an embodiment of the present invention, showing connectivity among the various components;

Brief Description of Drawings Paragraph:
[0034] FIG. 5 is a flow chart depicting an embodiment of the operation and control flow of instruction (executable code) and data pushing and client-server switching during the execution of an application program implemented using the process engine (i.e., network operating platform) of the present invention; and

Detail Description Paragraph:
[0038] The present invention relates to a system, method and computer program product for the secure creation and distribution of instructions to support network (e.g., Internet) application programs. The system, method and computer program product of the present invention is, in essence, a process engine (i.e., a network operating platform) that removes the structure of a client and a server in the network computational model.

Detail Description Paragraph:
[0044] Referring to FIG. 1B (and juxtaposing it to FIG. 1A), a block diagram illustrating a system architecture 150 utilized by an application service provider to offer application programs to users according to an embodiment of the present invention, is shown. Architecture 150 allows as user 102 to access the application program provided by the ASP via a communications network 112 (e.g., the Internet using TCP/IP, Fibre Channel Over IP (FCIP) or the like). Unlike architecture 100, however, the present invention provides a stable, robust and secure network

(Internet) operating platform (i.e., engine) 130 built directly on top of a network with individually addressable nodes which forms the complete infrastructure to provide the application program offered to the ASP's users (i.e., clients) 102.

Detail Description Paragraph:
[0045] Thus, in an embodiment of the present invention, an ASP is provided with the process engine (i.e., the network operating platform) described herein in order to facilitate its offering of one or more application programs to its clients (i.e., users) via an addressable network such as the Internet. The application program(s) may be an electronic auction, Internet search engine, entertainment or recreational game, business-to-business tool, business to consumer tool, chat service, e-mail service, personal financial service, information or news service, data retrieval service, product offering or the like. Further, such client use of the application program would be on a subscriber basis or a per-use basis, as applicable. That is, the ASP would provide the hardware (e.g., server machines) and the software (e.g., application software code logic and data stores) infrastructure to allow its customers to use the offered application programs(s).

Detail Description Paragraph:
[0046] Further, the engine of the present invention provides a means and method to allow a computational entity that performs the functions solely of a "client," to be immediately, altered to solely perform the functions of a "server," without regard as to the geographic, physical, or network topological location. It also provides the means and mechanism for a computational entity that performs the functions solely of a "server", to be immediately and without human intervention, altered to solely perform the functions of a "client", without regard as to the geographic, physical, or network topological location.

Detail Description Paragraph:
[0047] The net result of the engine 130 of the present invention is the effective control of the function of each computing node in the network, on a step-wise basis, for each and all processes. This control is not arbitrated by direct, immediately cognitive, human interaction. Rather, the contents of the data and application instructions which are present on the network, define, manage, and control the processes, based upon the logic stored or calculated within the engine 130.

Detail Description Paragraph:
[0049] In fact, after reading the following description, it will be apparent to one skilled in the relevant art(s) how to implement the following invention in alternative embodiments (e.g., networks other than the Internet, transport/network protocol suites other than TCP/IP (e.g., UDP/IP), and application programs other than those enumerated herein, and the like).

Detail Description Paragraph:
[0053] Referring to FIG. 2, a block diagram illustrating the physical architecture of a network (e.g., Internet) operating platform system 200, according to an embodiment of the present invention, is shown. FIG. 2 also shows connectivity among the various components of system 200. That is, system architecture 200 (architecture 150 shown in FIG. 1B) provides the stable, robust and secure engine 130 (as also shown in FIG. 1B) built directly on top of TCP/IP which forms the complete infrastructure to provide the application program offered to the ASP's users (i.e., clients) 102.

Detail Description Paragraph:
[0055] The users 102 would directly connect to the parts (i.e., infrastructure) of the system 200 which are provided by the ASP (i.e., elements 202-206 of FIG. 2) via the communications network (e.g., the Internet) 112. A dispatcher 202 is provided as the primary means to connect the ASP's infrastructure to the user 102. That is, in one embodiment of the present invention, a dispatcher machine 202 is connected

to network 112 and receives connection request from the users 102 of the network 112 to provide the services/functionality of the ASP's application program(s).

Detail Description Paragraph:
[0060] (204b) security data (e.g., list of "offending" network addresses which have previously tried to obtain unauthorized access to the ASP's application program(s), etc.);

Detail Description Paragraph:
[0064] (204f) user system data (e.g., hardware profile, network connection type, connected peripheral devices, etc. of each user 102).

Detail Description Paragraph:
[0068] Components 202, 204 and 206 of the system 200, as will be apparent to one skilled in the relevant art(s), are connected and communicate via a wide or local area network (WAN or LAN).

Detail Description Paragraph:
[0069] As used herein, "engine 130" refers to at least one dispatcher 202, a database 204 and a cluster of servant machines 206 that are physically interconnected and together operate to provide the functionality and advantages of the network operating platform described herein. Thus, as will be appreciated by one skilled in the relevant art(s), an ASP may offer its application program(s) to users 102 through one or more (mirrored) engines 130 within system 200.

Detail Description Paragraph:
[0073] In step 304, the user 102 establishes a network 112 connection (e.g., using TCP/IP) to the dispatcher 202 in order to utilize an application program offered by the ASP. In a preferred embodiment of the present invention, an ASP would be able to provide users 102 with an icon on the graphical user interface of the operating system they are utilizing (e.g., Microsoft.RTM. Windows.TM.) in order to establish this connection. The icon would, when clicked, for example, execute software code logic on the user machine to establish a connection to the network address (e.g., IP address) of the dispatcher 202 in a well-known manner.

Detail Description Paragraph:
[0075] Steps 306-312 reflect the fact, as suggested above, that system 200 may include multiple engines 130 (i.e., multiple dispatchers 202 each connected to different databases 204 that are, in turn, connected to a separate cluster of servant machines 206). That is, the ASP may utilize one or more separate engines 130 that operate in a distributed fashion. This allows, for example, a world-wide ASP to physically distribute data among its different resources (i.e., components 202, 204 and 206) for scalability and/or fault tolerance purposes. Further, this allows the ASP to mirror its engines 130 to make its application program(s) quickly available to local users or to reduce the load on heavily utilized engines 130.

Detail Description Paragraph:
[0076] Returning to FIG. 3, if the user 102 has reconnected to the same dispatcher 202 as before, control flow 300 proceeds to step 314. In steps 314 and 318, a security process determines whether the user 102 is actually the same user as before and then determines if they are authorized to access the network. This determination, in an embodiment of the present invention, is made by consulting database 204b (security data) to verify the source IP address of the user 102. In yet another embodiment of the present invention, the user 102 may have to provide a login and password, verified by accessing the administrative subscriber data within database 204e, to access the network.

Detail Description Paragraph:
[0077] If in step 306, the dispatcher 202 determines the user 102 has not been previously assigned a session, control flow 300 proceeds to step 316. In steps 316

and 321, a security process determines if the user 102 is authorized to access the application program. This determination, in an embodiment of the present invention, is made by the dispatcher 202 consulting database 204b (security data) to verify the source network address of the user 102. In yet another embodiment of the present invention, the user 102 may have to provide a login and password, verified by accessing data within database 204e--administrative subscriber data) to access the application program.

Detail Description Paragraph:
[0078] If in steps 318 or 321, the user is determined to be unauthorized to access the application program, control flow 300 proceeds to step 320. In step 320, the connection to the dispatcher 202 established by the user 102 is terminated and control flow 300 then ends as indicated by step 312. In an embodiment of the present invention, the termination event is logged to database 204 (i.e., security data) as an "offending" network address for future use as will be apparent to one skilled in the relevant art(s). In another embodiment of the present invention, the connection established by the user 102 session to the dispatcher 202 is not terminated, but the user is directed to a "demo" mode of the application program where no "real-world" data may be viewed, manipulated, or changed.

Detail Description Paragraph:
[0080] Once the availability determination is made, the dispatcher 202 assigns that specific servant machine 206 to the user 102 by supplying the user 102 with the network address of the servant machine 206 and instructions indicating how to establish a direct connection with the servant machine 206. The provision of connection instructions and data (i.e., IP address) to the user 102 is described in more detail below. At this point, the connection between the dispatcher 202 and user 102 is terminated. Then, in step 324, the user 102, utilizing the network address data and instructions received from the dispatcher 202, establishes a connection to the assigned servant machine 206. Control flow 300 then proceeds to step 326.

Detail Description Paragraph:
[0081] If in step 318, the user is determined to be authorized to access the network, the dispatcher 202 supplies the user 102 with the Network address of the servant machine 206 and instructions indicating how to establish a connection to the servant machine 206 as described in more detail above. Control flow 300 can then proceed to step 326.

Detail Description Paragraph:
[0082] In step 326, the application management process supplied by the engine 130 of the present invention commences. The application management process of step 326 is a dynamic process which provides the ASP's application program and specific resources to the user 102. That is, step 326 (and steps 328-332) represents the commencement of the execution of the application program on the servant machine 206 as it interacts (i.e., passes data and instructions back and forth) with the user 102. In an embodiment of the present invention, the user 102 may have to provide a login and password, verified by accessing database 204e (administrative subscriber data), to access the network. In yet another embodiment of the present invention, in step 326, the user may be presented with a choice of multiple application programs offered by the ASP.

Detail Description Paragraph:
[0086] The application management process of step 326 also includes a dynamic routing process that iteratively calculates the average throughput of a user 102 session while connected to the engine 130, and has the ability during the session to place users 102 who are working slowly, onto slower connections ("skinny pipes"), by re-assigning the network connection location. This re-assignment does not require the session to stop. The dynanic routing process also enables a "lost" or "dropped" connection to be resumed, without knowledge or interaction by the user

102. This is accomplished by the engine 130 storing the details of each socket connection, and utilizing a routing table with information concerning the connecting of the two sockets. It simply re-opens the path between the two sockets, and informs the user kernel that the socket connection has been reconnected.

Detail Description Paragraph:
[0088] Step 328 reflects the fact, as suggested above, that system 200 may include multiple engines 130 (i.e., dispatchers 202 each connected to different databases 204 which are connected to separate cluster of servant machines 206). That is, the ASP may utilize one or more separate engines 130 that operate in a distributed fashion within system 200. This allows, for example, a world-wide ASP to physically distribute data among its different resources (i.e., components 202, 204 and 206) for scalability and/or fault tolerance purposes. Further, this allows the ASP to mirror its engines 130 to make its application program(s) quickly available to local users or to reduce the load on heavily utilized engines 130.

Detail Description Paragraph:
[0095] More specifically, the computational resources database 204a contains active information specifics as to which nodes with computational abilities are presently connected to a specific engine 130, exactly where those computational resources exist (according to logical location, as through a network address or other digital location), where all routing gateways may exist, what devices are on what domains, system latency between all devices, etc. The database 204a is also kept aware of the existence of other engines 130, at other network locations, that can perform some or all of the same functions. For example, database 204a contains the name, address, processor power rating, available storage and work space, available task queues, and present load factor for every device connected to system 200. As each event changes the status of facts regarding what devices are connected, or where they are, the database 204ais updated. The database values are stored, after active polling of each resource by the engine 130 code logic, and actively indicates the status of information in real-time.

Detail Description Paragraph:
[0098] Thus, given the mnemonics of TABLE 1, an exemplary servant machine 206 availability determination may involve any computation involving these mnemonics. For example, the current utilization load of any resource within system 200 may be calculated by arithmetically calculating a resulting variable that is derived from the following factors: length of the queue of commands waiting to be processed by each computational resource, the computational power of each resource, the complexity of each of the queued commands for that resource, and the network latency of each computational resource.

Detail Description Paragraph:
[0105] Referring to FIG. 4, a block diagram of a software architecture 400 of the part of the process engine (i.e., network operating platform) 130 that resides on the servant machines 206, according to an embodiment of the present invention, is shown. FIG. 4 also shows connectivity among the various components of architecture 400. Software architecture 400 reflects that part of the code logic that comprises engine 130 which resides on the servant machines 206. (The other portions of the code logic reside on the dispatcher 202 and database 204.) Thus, FIG. 4 illustrates a servant machine 206 which includes a plurality of S-bot processes 402 (shown as S-bot processes 402a-n) and a central C-bot process 404.

Detail Description Paragraph:
[0107] In a preferred embodiment, the software code logic implementing each of the S-bot processes is multi-threaded. That is, the program execution environment interleaves instructions from multiple independent execution "threads." The multi-threaded S-bot processes 402 thus allow multiple instances of each component (thread) to run simultaneously thereby increasing the throughput of the system 200 as a whole. As will be apparent to one skilled in the relevant art(s), each thread

comprising an S-bot process 402 is responsible for performing a specific task within the specific application program (e.g., calculating the dot product of a matrix, calculating the value of .pi. to a certain level of precision, etc.). Consequently, each thread within an S-bot process 402 is identified by a network address (i.e., the network address of the servant machine 206 it is executing on), a port number, and a unique socket number.

Detail Description Paragraph:
[0114] As suggested above, during control flow 300 (i.e., step 322), the dispatcher 202 determines which specific servant machine 206 is available (i.e., which servant machine has the available computation resources in order to service the user 102 session and provide the application program). Then, the dispatcher assigns that specific servant machine 206 to the user 102 by supplying the user 102 with the network address of the servant machine 206 (i.e., data) and instructions (i.e., executable code) indicating how to establish a connection to the servant machine 206.

Detail Description Paragraph:
[0115] As also suggested above, during the execution of the application program (i.e., the application management process of step 326 and steps 328-332), the servant machine 206 receives queries from the user, parses them, and determines what resources are required to respond to such queries. The provision of the network to the user 102 and its execution of an application program involves the exchange of data and instructions (i.e., executable code) between the user 102 machine and the servant machine 206 that switches their respective roles as "client" or "server."

Detail Description Paragraph:
[0116] In the above two instances (i.e., step 322 and steps 326-332), instructions and data are pushed to and from one of the servant machines 206 and the user 102 machine. This is achieved by the "pushing" back and forth of instructions (and associated data) derived from computed logic without any knowledge by, or interaction from, the user 102. Answers to queries may be "pushed" between any entities connected to the network, or may be stored for the completion of "push" delivery, when a device which is temporarily unavailable, becomes available again to the network.

Detail Description Paragraph:
[0118] In sum, the engine 130 allows application instructions to be passed from one point to another in an optimized format. These instructions are then executed in a location that may be more efficient for the process at hand. The net result is the effective control of the function of each computing component in the network, on a step by step basis. Thus, the contents of the data and application instructions which are present on the network, define, manage and control the processes. Further, engine 130 allows a computational entity (i.e., user 102 machine processes) that conventionally performs the functions solely of a "client," to be immediately and without human intervention, altered to perform the functions of a "server." It also allows a computational entity (i.e., servant machine 206 processes) that conventionally performs the functions solely of a "server," to be immediately and without human intervention, altered to perform the functions of a "client." In every case, the former "client" that becomes a "server," has no legacy knowledge of its former role.

Detail Description Paragraph:
[0124] Thus, in step 510, when the user 102 forms another query, the data responsive to that subsequent query may already reside on the user 102 machine, as determined by step 512-thus making it a server. The instructions sent to the user 102 machine "teaches" it how to behave as a server, or how to perform operations that would not be a part of its native intelligence or instruction repertoire, and use the one or more data pages received to answer its own queries. If the data are

not available (i.e., it is not part of the page of data sent in step 506), the servant machine reverts back to its role as a "server" and thus, the user 102 machine revert back to its role as a "client" in step 514. As a consequence of the above, system 200 of the present invention (and more specifically, engine 130) can function as one of a category of applications known as "browsers", with the important exceptions that it does not require pre-defined formats for the presentation of any page, segment, unit, or quantity of information, and that it requires neither regard as to the information type or category, nor regard to the native ability to execute HTML or other similar page description languages. Therefore, one embodiment of the engine 130 of the present invention functions as a network operating platform for ASP's to offer application programs or other services to their clients.

Detail Description Paragraph:
[0134] As will be apparent to one skilled in the relevant art(s) after reading the description herein, the user 102 machine would need to contain a kernel that would allow it to execute the instructions received from the servant machines 206 in the two instances described above--the dispatcher assignment of step 322, and the application management process of steps 326-332. As will be appreciated by one skilled in the relevant art(s), a kernel is the part of an operating system--in this case the Internet operating platform of the present invention--that is responsible for resource allocation, low-level hardware interfaces, security, etc. In a preferred embodiment of the present invention, as suggested above, the kernel would be supplied by the ASP to the user 102 and be accessed by means of a connection to an icon on the graphical user interface of the operating system that the user is utilizing in order to initially establish the connection to the network address of the dispatcher 202.

Detail Description Paragraph:
[0135] The customer query is first passed in parsed format to the engine 130. This parsing is achieved by a parsing module on the customer machine that is part of the kernel that resides on the user 102 computer. This parsing module translates the customer query into a format directly executable by the engine 130. As a result, only optimized code is transferred between the user 102 and the engine 130. If the parsing commences an iterative action, the engine 130 sends similarly parsed code back to the kernel. The kernel also provides the ability to interpret (and execute) this optimized parsed code on the machine which contains the kernel. In an embodiment of the present invention this bi-directional parsed code is transmitted in the form of symbols. Each ASP may utilize a (proprietary) symbol library for any session which includes both a collection of elementary (i.e., base) symbols and functions related thereto depending on the specific network management, context, and application program being provided by the ASP. Further, a dynamic extension to the base library, which is unique to a particular user 102 session can direct a user 102 machine to execute tasks which are unknown to any other kernel (i.e., user 102) connected to the system 200. This results in a system 200 where each of a multiplicity of diverse and unique users 102 receive the specific resources and capabilities they require to resolve their individual queries, in whatever format they require.

Detail Description Paragraph:
[0136] In an embodiment of the present invention, data and code packets are transmitted intermixed, as strings of symbols or ASCII characters. The engine 130 and the kernel know how to parse these strings to separate code from data and one instruction from another. For example, the dispatcher 202 may send the digital information "123.456.789.0:1234" to the kernel on the user machine 102, which would direct the kernel to establish a socket-to socket communications session with a location having a network address of 123.456.789.0, using port number 1234. Strings need not be transmitted in sequence, can be packed so as to minimize the number of packets required for network transmission, and can be encrypted so as to secure their contents.

Detail Description Paragraph:
[0140] The present invention provides a system, method and computer program product for the secure creation and distribution of instructions to support network (e.g., Internet) applications. Thus, various security measures within system 200 are now addressed.

Detail Description Paragraph:
[0141] Components 202, 204 and 206 of system 200, as will be apparent to one skilled in the relevant art(s), are connected and communicate via a wide or local area network (WAN or LAN) running a secure communications protocol (e.g., secure sockets layer (SSL)) layered above the connection protocol (e.g., TCP/IP). Further, components 202, 204 and 206 of system 200 may be protected by a firewall (not shown in FIG. 2). Generally speaking, a firewall is a dedicated gateway machine (e.g., a SUN Ultra 10) with special security precaution software. It is typically used, for example, to service network 112 connections and dial-in lines, and protects the cluster of more loosely administered network elements hidden behind it, from external invasion. Firewalls are well known in the relevant art(s) and firewall software is available from many vendors such as Check Point Software Technologies Corp. of Redwood City, Calif.

Detail Description Paragraph:
[0142] In an embodiment of the present invention, instructions and data pushed to and from one of the servant machines 206 and the user 102 machine would be encrypted for security purposes. That is, such instructions and data would first be encrypted before transmitting them using the protocol (e.g., TCP/IP) implemented in the network (e.g., the Internet). This would allow the security needs of the application program (e.g., a product ordering application program that involves transmitting user 102 credit card information for payment purposes) to be met. Such encryption could utilize any known cryptography method to prevent any but the intended recipient and author from reading the transmitted instructions and associated data (e.g., RSA public-key encryption, Data Encryption Standard (DES) and the like). Further, a lower bit encryption (e.g., 40-bit or 256-bit encryption) would be used for "normal" (i.e., chat, messaging, or other non-payment) data and a higher bit encryption (e.g., 1024-bit encryption) would be used for any accounting, payment or other financial-type data. A detailed discussion of cryptography can be found in Bruce Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C," John Wiley & Sons, 2nd ed., ISBN 0-47-112845-7(USA 1996), which is incorporated herein by reference in its entirety.

Detail Description Paragraph:
[0146] In an embodiment of the present invention, the security process (steps 316-321 of control flow 300) determines whether the network location of the originator of the "first request for service" is contained in a list of "security failures" contained in the security database 204b. It also checks for validity in the connection method (making sure that the user 102 is attempting to connect to a specific location (network IP address+port address) that is authorized by the system 200), protocols, packet contents, and searches for anomalies in data structure and content (i.e., the library of symbols use for interpretation of parsed code pushed back and forth as explained herein).

Detail Description Paragraph:
[0151] The present invention (i.e., engine 130, system 200, control flow 300, architecture 400, control flow 500 or any part thereof) may be implemented using hardware, software or a combination thereof and may be implemented in one or more computer systems or other processing systems. In fact, in one embodiment, the invention is directed toward one or more computer systems capable of carrying out the functionality described herein. An example of a computer system 600 is shown in FIG. 6. The computer system 600 includes one or more processors, such as processor 604. The processor 604 is connected to a communication infrastructure 606 (e.g., a

communications bus, cross-over bar, or network). Various software embodiments are described in terms of this exemplary computer system. After reading this description, it will become apparent to a person skilled in the relevant art(s) how to implement the invention using other computer systems and/or computer architectures.

Detail Description Paragraph:
[0155] Computer system 600 may also include a communications interface 624. Communications interface 624 allows software and data to be transferred between computer system 600 and external devices. Examples of communications interface 624 may include a modem, a network interface (such as an Ethernet card), a communications port, a PCMCIA slot and card, etc. Software and data transferred via communications interface 624 are in the form of signals 628 which may be electronic, electromagnetic, optical or other signals capable of being received by communications interface 624. These signals 628 are provided to communications interface 624 via a communications path (i.e., channel) 626. This channel 626 carries signals 628 and may be implemented using wire or cable, fiber optics, a phone line, a cellular phone link, an RF link and other communications channels.

Detail Description Table CWU:
2TABLE 1 MNEMONIC DESCRIPTION AU.sub.tcE.sub.n Number of Active Users connected to Engine n E.sub.n Identifies the Engine number, in the array of on-line engines E.sub.nanEP calculated as Number of Available Network Entry Points on Engine n (E.sub.nt.sub.nEP-E.sub.no.sub.nEP) E.sub.nC.sub.n Identifies the Engine computer number (from 1 to n) that is online in the engine E.sub.nC.sub.nP.sub.n Identifies Engine n Computer number (from 1 to n) and Process number (from 1 to n) that is online in the engine E.sub.nC.sub.nP.sub.nQ.- sub.n Identifies the Engine number, Computer number, Process number, and Task Queue number (from 1 to n) that is online in the engine E.sub.nC.sub.nP.sub.nQ.sub.nTL.sub.n Identifies the Engine number, Computer number, Process number, Queue number, and Task List size for a task queue E.sub.nCR.sub.nCP Calculated as Engine n, Computing Resource n, raw Computing Power E.sub.nCR.sub.nP.sub.n Engine n, Computing Resource n, number of Processes n E.sub.nO.sub.nEP Number of Occupied Network Entry Points on Engine n E.sub.nSWV Defined as the Switch Weighting Value for Engine n E.sub.nTLL Calculated as the present threshold limit load of Engine n (when this value exceeds E.sub.nTLV, the engine is no longer able to accept commence new sessions E.sub.nTLH Calculated as the Threshold Limit headroom for Engine n E.sub.nTLV Calculated or stated as the threshold limit value of Engine n (calculations producing values above this value, cause Dispatcher to re-direct the user attempting to connect, to a different engine) E.sub.ntnEP Number of Total Network Entry Points on Engine n L.sub.cE.sub.nU.sub.n Calculated as Latency, (Communication) between Engine n and User n (L.sub.tnE.sub.nU.sub.n + L.sub.taE.sub.n) (Network + Engine, 1 way to user) L.sub.taE.sub.n Calculated as Latency, (Total Assembly and Aggrega- tion) for Engine n (Compression, Encryption, etc.) (Host Engine only) L.sub.tcU.sub.nE.sub.n Calculated as Latency, (Total Communication) between Engine n and User n (Network + Engine, User to Host) L.sub.tnE.sub.nE.sub.n Calculated as Latency, (Total Network) between two different Engines L.sub.tnE.sub.nU.sub.n calculated as Latency, (Total Network), between Engine n and User n (Network, 1-way) L.sub.trcE.sub.nU.sub.n Calculated as Latency (Total Return Communication) between Engine n and User n (Network + Engine, return) L.sub.trE.sub.nE.sub.n Calculated as Latency, (Total Network), between Engine n and Engine n (Network, 2-way) L.sub.trE.sub.nU.sub.n calculated as Latency, (Total Return) between Engine n and User n (Network, 2-way) P.sub.tE.sub.n Total Number of Processes running on Engine n SU.sub.tcE.sub.n Number of Stalled Users connected to Engine n U.sub.nCP Calculated as the User n Computing Power, it is an index of how powerful the user computer is. It is derived from the time required for the User n computer to produce an answer to a standard mathematical question. U.sub.nTP Calculated as the data throughput of User n, by taking the amount of data received from User n, and dividing by a time unit U.sub.tc Calculated as the total number of users connected to all PB Engines (Sum of U.sub.tecE.sub.1 to U.sub.tcE.sub.n) U.sub.tecE.sub.n Calculated as Total Number

of Users connected to Engine n (AU.sub.tcE.sub.n + SU.sub.tcE.sub.n)

Detail Description Table CWU:
3TABLE 2 FIELD DESCRIPTION YlwFlag0 List of network addresses of prematurely-terminated sessions Addresses (i.e. dropped connections) YlwFlag1 network addresses of originators, who have attempted to Addresses connect to incorrect ports; # of attempts at each port YlwFlag2 list if network addresses being thoroughly logged for Addresses suspicious activity YlwFlag3 list of suspicious character strings, etc. Pre-cursor to virus Strings detection, hack attempts RedFlag1 list of network addresses that have attempted "denial of Addresses service" attacks RedFlag2 list of network addresses that have attempted unauthorized Address uploads OrgFlag1 List of network addresses requiring contact with a customer Addresses service representative OrgFlag2 List of network addresses requiring immediate suspension for Addresses cause GreyFlag1 List of network addresses with service suspended by request Addresses of user VltFlag1- List of network addresses under legal surveillance VltFlagn Addresses

CLAIMS:

1. A method for the secure creation and distribution of instructions to support a network application, comprising the steps of: (1) receiving a connection request from a user machine; (2) selecting one of a plurality of servant nodes; (3) sending a first code package to said user machine, wherein said first code package includes instructions and data that facilitate said user machine connecting to said one of said plurality of servant nodes; (4) establishing a connection between said user machine and said one of said plurality of servant nodes; (5) commencing an application on said one of said plurality of servant nodes; (6) receiving a query from said user, wherein said query is related to said application; (7) sending a second code package to said user machine, wherein said second code package includes instructions and data partially responsive to said query; (8) receiving an answer computed by said user machine utilizing said second code package; (9) parsing said answer received from said user machine and combining said answer with additional data to form a final response to said query received from said user machine; and (10) sending said user said final response; whereby said first and second code packages allow an application service provider to support the delivery of a network-based application program that is uniquely presented, structured, and executed for said user.

2. A system for providing an application service provider with a network-based operating platform to deliver application programs to a user, comprising: (A) a dispatcher connected over a network to at least one user machine, wherein said dispatcher has means for receiving a connection request from said at least one user machine; (B) a plurality of servant nodes; (C) a database connected to said dispatcher and to each of said plurality of servant nodes; (D) means for said dispatcher to access said database in order to select one of said plurality of servant nodes that is available to service said connection request from said at least one user machine; (E) means, within said a dispatcher, for sending a first code package to said at least one user machine, wherein said first code package includes instructions and data that facilitate its connection to said one of said plurality of servant nodes; and (F) means for said one of said plurality of servant nodes to access said database in order to send a second code package to said at least one user machine, wherein said second code package includes instructions and data responsive to query received from said at least one user machine; whereby said second code package allows the application service provider to deliver the network-based application program to the user.

3. The system of claim 2, wherein said network is at least a portion of the Internet.

4. A computer program product comprising a computer usable medium having control

logic stored therein for causing a computer to provide the secure creation and distribution of instructions to support a network application, said control logic comprising: first computer readable program code means for causing the computer to receive a connection request from a user machine; second computer readable program code means for causing the computer to select one of a plurality of servant nodes; third computer readable program code means for causing the computer to send a first code package to said user machine, wherein said first code package includes instructions and data that facilitate said user machine connecting to said one of said plurality of servant nodes; fourth computer readable program code means for causing the computer to establish a connection between said user machine and said one of said plurality of servant nodes; fifth computer readable program code means for causing the computer to commence an application on said one of said plurality of servant nodes; sixth computer readable program code means for causing the computer to receiving a query from said user, wherein said query is related to said application; seventh computer readable program code means for causing the computer to send a second code package to said user machine, wherein said second code package includes instructions and data partially responsive to said query; eighth computer readable program code means for causing the computer to receive an answer computed by said user machine utilizing said second code package; ninth computer readable program code means for causing the computer to parse said answer received from said user machine and combining said answer with additional data to form a final response to said query received from said user machine; and tenth computer readable program code means for causing the computer to sending said user said final response.

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L40: Entry 27 of 27 File: USPT Oct 14, 1997

US-PAT-NO: 5678061
DOCUMENT-IDENTIFIER: US 5678061 A

TITLE: Method for employing doubly striped mirroring of data and reassigning data streams scheduled to be supplied by failed disk to respective ones of remaining disks

DATE-ISSUED: October 14, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mourad; Antoine N.	Aberdeen	NJ		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Lucent Technologies Inc.	Murray Hill	NJ			02

APPL-NO: 08/ 504096 [PALM]
DATE FILED: July 19, 1995

INT-CL: [06] G06 F 15/02

US-CL-ISSUED: 395/841; 395/182.04, 395/405
US-CL-CURRENT: 710/21; 711/5, 714/6

FIELD-OF-SEARCH: 395/182.04, 395/182.05, 395/441, 395/439, 395/489, 395/841, 395/405, 395/497.04, 395/800, 395/842, 395/182.03, 371/40.4

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4688168	August 1987	Gudaitis et al.	395/287
☐	4796098	January 1989	Giddings	386/47
☐	4849978	July 1989	Dishon et al.	395/182.04
☐	5088081	February 1992	Farr	369/54
☐	5235601	August 1993	Stallmo et al.	371/40.1
☐	5303244	April 1994	Watson	395/182.03

☐	5404454	April 1995	Parks	395/841
☐	5463758	October 1995	Ottesen	395/441
☐	5471640	November 1995	McBride	395/842
☐	5487160	January 1996	Bemis	395/441
☐	5497478	March 1996	Murata	395/484
☐	5499337	March 1996	Gordon	395/182.04
☐	5519435	May 1996	Anderson	395/441
☐	5519844	May 1996	Stallmo	395/441

OTHER PUBLICATIONS

Third Intl Workshop, Nov. 1992, "The Design and Implementation of a Continuous Media Storage Server", P. Lougher et al, pp. 69-80.
Proceedings of 26th Hawaii Intl Conf. on System Sciences, v. 1, 1993 IEEE "Disk Subsystem Load Balancing: Disk Striping vs. Conventional Data Placement", G. R. Ganger et al, pp. 40-49.
"Performance Analysis of a Dual Striping Strategy for Replicated Disk Arrays", by Merchant et al, 1993, pp. 148-157.
"A Synchronous Disk Interleaving", by Kim et al, 1991 IEEE, pp. 801-810.
"Analytic Modeling and Comparisons of striping Strategies for Replicated Disk Arrays", by Merchant et al, IEEE 1995, pp. 419-433.
"Chained Declustering", by Hsiao et al, IEEE 1990, pp. 456-465.
"Tuning of Striping Units in Disk--Array--Based File System", by Weikum et al, IEEE 1992, pp. 80-87.
"An Approach to Cost--Effective Terabyte Memory Systems", by Katz et al, IEEE 1992, pp. 395-400.
"A Performance Study of Three High Availabilty Data Replication Strategies" by Hsiao et al, IEEE 1991, pp. 18-28.
"Replicated Data Management in the Gamma Database Machine", by Hsiao et al, 1990 IEEE, pp. 79-84.
"Introduction to Redundant Arrays of Inexpensive Disks (RAID)", by Patterson et al, IEEE 1989, pp. 112-117.
"Disk Subsystem Load Balancing", by Ganger et al, IEEE 1993, pp. 40-49.
"Chained Declustering", by Golubchik et al, IEEE 1992, pp. 88-95.
"Systems Reliability and Availability Prediction", by Daya Perera, IEEE 1993, pp. 33-40.
"Communications--Intensive Workstations", by Katseff et al, IEEE 1992, pp. 24-28.
"LAN Lirchin", by Harbison, Robert, Feb. 1994, LAN Magazine, v9, n2, p. 93(14).
"Diamonds are Forever", by Radding, Alan, Midrance Systems, Feb. 9, 1993, v6, n3, p. 23(2).

ART-UNIT: 237

PRIMARY-EXAMINER: Meky; Moustafa M.

ATTY-AGENT-FIRM: Luludis; F. B.

ABSTRACT:

The reliability of supplying data stored in a plurality of different memories to different users is enhanced by (a) dividing each of the memories into primary and secondary sections, (b) partitioning the data into successive blocks and (c) storing the blocks of data in sequence in respective ones of the primary sections. Then storing in sequence the blocks of data that have been stored in the primary

section of one of the memories in respective ones of the secondary sections of the other ones of said disks.

2 Claims, 8 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L40: Entry 26 of 27 File: USPT Dec 7, 1999

US-PAT-NO: 6000020
DOCUMENT-IDENTIFIER: US 6000020 A

TITLE: Hierarchical storage management from a mirrored file system on a storage network segmented by a bridge

DATE-ISSUED: December 7, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chin; Howey Q.	San Jose	CA		
Chan; Kurt	Roseville	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Gadzoox Networks, Inc.					02

APPL-NO: 08/ 825683 [PALM]
DATE FILED: April 1, 1997

INT-CL: [06] G06 F 12/16, G06 F 13/00, H04 J 3/02

US-CL-ISSUED: 711/162; 711/111, 711/112, 711/114, 709/214, 709/216, 709/249, 370/401, 370/403, 370/404, 370/405
US-CL-CURRENT: 711/162; 370/401, 370/403, 370/404, 370/405, 709/214, 709/216, 709/249, 711/111, 711/112, 711/114

FIELD-OF-SEARCH: 395/200.44, 395/200.45, 395/200.46, 395/200.79, 395/182.02, 395/182.03, 395/182.04, 370/401-409, 711/111-114, 711/162, 709/214, 709/216, 709/249

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5179555	January 1993	Videlock et al.	370/85.13
☐	5212784	May 1993	Sparks	395/575
☐	5432907	July 1995	Picazo, Jr. et al.	395/200
☐	5488731	January 1996	Mendelsohn	395/182.04
☐	5673382	September 1997	Cannon et al.	395/182.04

☐	5694615	December 1997	Thapar et al.	395/180
☐	5757642	May 1998	Jones	395/200.33
☐	5771349	June 1998	Picazo, Jr. et al.	395/188.01
☐	5831985	November 1998	Sandorfi	370/468
☐	5848251	December 1998	Lomelino et al.	710/12P

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
A2 0 359 471	September 1989	EP	
WO 94/25919	November 1994	WO	
WO 96/42019	December 1996	WO	

ART-UNIT: 272

PRIMARY-EXAMINER: Thai; Tuan V.

ATTY-AGENT-FIRM: Fish; Ronald C. Falk & Fish

ABSTRACT:

A system for hierarchical data storage management and transparent data backup in a high speed, high volume Fibre Channel Arbitrated Loop environment comprising first and second Fibre Channel Arbitrated Loops, each coupling a Transaction Server and backup HSM server to high speed disk drives and mirrored high speed disk drives respectively. The two loops are coupled by a Bridge compatible with the Fibre Channel Arbitrated Loop protocol which forwards write transactions directed to the mirrored disk drives from the first loop the second but keeps read transaction from the Transaction Server to the high speed disk drives on the first loop isolated from backup and HSM transactions occurring on the second loop between the backup HSM server, the mirrored disk drives and backup storage devices coupled to the backup HSM server.

13 Claims, 18 Drawing figures